TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission	15

Application Number	**10/007,766**
Filing Date	**November 8, 2001**
First Named Inventor	**Brad R. LEWIS et al.**
Group Art Unit	2191
Examiner Name	Qamrun Nahar
Attorney Docket Number	30014200-1002

ENCLOSURES *(check all that apply)*

☒ Transmitted herewith is a Response to Office Action of October 17, 2006.

☒ The fee has been calculated as shown below:

(1) FOR	(2) CLAIMS REMAINING AFTER AMENDMENT	(3)	(4) HIGHEST NO. PREVIOUSLY PAID FOR	(5) PRESENT EXTRA	(6) RATE	(7) ADDITIONAL FEE
TOTAL CLAIMS	26	-	26	0	☐ x $25.00 ☒ x $50.00	$0.00
INDEPENDENT CLAIMS	7	-	7	0	☐ x $100.00 ☒ x $200.00	$0.00
	APPLICATION AMENDED TO CONTAIN ANY MULTIPLE DEPENDENT CLAIMS NOT PREVIOUSLY PAID FOR.			☐ YES ☒ NO	☐ x $180.00 ☒ x $360.00 ONE TIME	$0.00
				TOTAL ADDITIONAL FEE FOR THIS AMENDMENT		$0.00

☒ Applicant petitions the Commissioner for Patents to extend the time for responding to the Office Action dated October 17, 2006 by three month(s) for a fee of $1,020.00 so that the period for response is extended to April 17, 2007 under 37 C.F.R. § 1.136.

☐ The amount of $_____ for the Terminal Disclaimer under 37 C.F.R. § 1.20(d) is included in the enclosed credit card payment form to charge.

☒ The enclosed credit card payment form to charge the amount of $1,020.00 is to cover the total claim fee and other applicable fees.

☒ The Commissioner is hereby authorized to charge the extension fee and any additional fees which may be required, or to credit any overpayment to Account No. 19-3140. **A duplicate of this sheet is enclosed.**

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

14. ☒ Customer No. 58328

Dated: April 6, 2007

A. Wesley Ferrebee (Registration No. 51,312)

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date indicated below.

Dated: _____

_____

Attorney Docket No. 30014200-1002 PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:	)	Group Art Unit: 2191
	)	
Brad R. LEWIS et al.	)	Examiner: Qamrun Nahar
	)	
Application No. 10/007,766	)	
	)	
Filed: November 8, 2001	)	
	)	
For: Methods and Systems for Developing Data Flow Programs	)	

MAIL STOP Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

RESPONSE TO OFFICE ACTION OF OCTOBER 17, 2006

Dear Sir:

This Response is submitted in response to the Office Action mailed October 17, 2006. Applicants respectfully request reconsideration and allowance of the pending claims.

IN THE CLAIMS

This listing of claims replaces all prior listings:

1. (Previously Presented) A method in a data processing system for developing a data flow program comprising code segments that operate on data in memory, the method comprising the steps of:

dividing the memory into blocks;

assigning at least a portion of the data and at least one code segment to each block;

storing data read and data write identifiers for each code segment, the data read and data write identifiers identifying at least a portion of the data read or written by the code segment;

determining whether dependencies exist among the blocks such that a first block depends on data assigned to a second block using the read and write identifiers; and

facilitating development of the data flow program by generating a graph representing the blocks and the determined dependencies and displaying the graph to a user.

2. (Original) A method according to claim 1, wherein the step of displaying comprises the step of displaying a graph comprising nodes assigned to the blocks and dependency arcs representing the determined dependencies.

3. (Original) A method according to claim 2, wherein the step of displaying further comprises the step of presenting the dependency arcs using a satisfied dependency visualization when the determined dependency is satisfied, and presenting the dependency arcs using an unsatisfied dependency visualization when the determined dependency is unsatisfied.

4. (Original) A method according to claim 2, further comprising the steps of:

receiving a node selection specifying a selected one of the nodes;

determining unmet dependencies for the selected node; and

displaying in a visually distinctive manner the unmet dependencies in the graph.

5. (Original) A method according to claim 2, further comprising the steps of:

providing for execution of the code segments using threads;

receiving a thread selection specifying at least one of the threads; and

displaying nodes executed by the at least one thread.

6. (Previously Presented) A method according to claim 2, wherein the nodes include executed nodes and unexecuted nodes, and wherein the step of displaying further comprises the step of displaying the unexecuted nodes using an unexecuted visualization and the executed nodes using an executed visualization.

7. (Original) A method according to claim 1, wherein the data includes a data structure, and wherein the step of displaying further comprises the step of:

facilitating visualization of at least a portion of the data structure accessed by at least one of the code segments by graphically presenting at least a portion of the data structure and accentuating the portion of the data structure accessed by the at least one code segment.

8. (Previously Presented) A method in a data processing system for developing a data flow program comprising code segments distributed between memory blocks, the method comprising the steps of:

representing the data flow program as a graph comprising nodes and node dependencies between the nodes; and

displaying the graph to facilitate visualization of the data flow program,

wherein the node dependencies between nodes are determined based on data read and data write identifiers for code segments associates with the respective nodes, the data read and data write identifiers identifying at least a portion of data read or written by the respective code segment.

9. (Original) A method according to claim 8, wherein the nodes include executed nodes and unexecuted nodes, and wherein the step of displaying comprises the step of displaying the unexecuted nodes with an unexecuted visualization and displaying the executed nodes with an executed visualization.

10. (Original) A method according to claim 9, wherein the nodes include executing nodes, and wherein the step of displaying comprises the step of displaying the executing nodes with an executing visualization.

11. (Original) A method according to claim 8, wherein the node dependencies include satisfied dependencies and unsatisfied dependencies, and wherein the step of displaying comprises the steps of displaying the unsatisfied dependencies using an unsatisfied dependency

visualization, and displaying the satisfied dependencies using a satisfied dependency visualization.

12. (Previously Presented) A computer-readable medium containing instructions that cause a data processing system to perform a method for developing a data flow program comprising code segments that operate on data in memory, the method comprising the steps of:

dividing the memory into blocks;

assigning at least a portion of the data and at least one code segment to each block;

storing data read and data write identifiers for each code segment, the data read and data write identifiers identifying at least a portion of the data read or written by the code segment;

determining a dependency imparted by a first block depending on data assigned to a second block using the read and write identifiers; and

facilitating development of the data flow program by generating a graph representing the blocks and the determined dependencies and displaying the graph to a user.

13. (Original) A computer-readable medium according to claim 12, wherein the step of displaying comprises the step of displaying a graph comprising nodes assigned to the blocks and a dependency arc representing the determined dependency.

14. (Previously Presented) A computer-readable medium according to claim 13, wherein the step of displaying further comprises the step of presenting the dependency arc using a satisfied dependency visualization when the determined dependency is satisfied, and presenting

the dependency arc using an unsatisfied dependency visualization when the determined dependency is unsatisfied.

15. (Original) A computer-readable medium according to claim 13, further comprising the steps of:

receiving a node selection specifying a selected node;

determining unmet dependencies for the selected node; and

highlighting in the graph the unmet dependencies.

16. (Original) A computer-readable medium according to claim 13, further comprising the steps of:

providing for execution of the code segments using threads;

receiving a thread selection specifying at least one of the threads; and

displaying nodes executed by the at least one thread.

17. (Previously Presented) A computer-readable medium according to claim 13, wherein the nodes include executed nodes and unexecuted nodes, and wherein the step of displaying further comprises the step of presenting the unexecuted nodes using an unexecuted visualization and the executed nodes using an executed visualization.

18. (Original) A computer-readable medium according to claim 12, wherein the data includes a data structure, and wherein the step of displaying further comprises the step of:

facilitating visualization of at least a portion of the data structure accessed by at least one of the code segments by graphically presenting at least a portion of the data structure and accentuating the portion of the data structure accessed by the at least one code segment.

19. (Original) A method in a data processing system for developing a data flow program comprising code segments that operate on data in a memory, the method comprising the steps of:

dividing into blocks the memory that stores the data;

for each block, assigning at least a portion of the data to the block and assigning at least one of the code segments to the block;

storing data read and data write identifiers for each code segment, the data read and data write identifiers identifying at least a portion of the data read or written by the code segment;

determining whether dependencies exist among the blocks such that a first block depends on data assigned to a second block using the read and write identifiers;

generating a directed acyclic graph comprising nodes and arcs between the nodes by assigning the blocks to the nodes and by assigning the dependencies to the arcs;

displaying the directed acyclic graph;

initiating execution of the code segments;

while the code segments are executing,

determining which nodes in the graph are unexecuted nodes and which nodes in the graph are executed nodes; and

displaying the unexecuted nodes in a manner visually distinctive from the executed nodes.

20. (Previously Presented) A data processing system comprising:

a memory comprising a data flow program and a data flow development tool that associates data processed by the data flow program to blocks in the memory, associates code segments of the data flow program to at least one of the blocks, stores data read and data write identifiers for each code segment, the data read and data write identifiers identifying at least a portion of the data read or written by the code segment, determines dependencies between the blocks using the read and write identifiers, and displays a graph comprising nodes and arcs depicting the dependencies between the blocks; and

a processor that runs the data flow development tool.

21. (Original) The data processing system of claim 20, wherein the nodes comprise executed nodes and unexecuted nodes, and wherein the executed nodes are displayed using an executed node visualization and the unexecuted nodes are displayed using an unexecuted node visualization.

22. (Original) The data processing system of claim 20, wherein the arcs comprise satisfied dependency arcs and unsatisfied dependency arcs, and wherein the satisfied dependency arcs are displayed using a satisfied dependency visualization and the unsatisfied dependency arcs are displayed using an unsatisfied dependency visualization.

23. (Previously Presented) A data processing system for developing a data flow program comprising code segments that operate on data in memory, the data processing system comprising:

means for apportioning a memory into regions and associating the data and the code segments with the regions;

means for storing data read and data write identifiers for each code segment, the data read and data write identifiers identifying at least a portion of the data read or written by the code segment;

means for determining dependencies between the regions using the data read and write identifiers; and

means for displaying a graph of nodes that are assigned regions, and arcs depicting the dependencies between the regions.

24. (Previously Presented) A computer readable memory device encoded with a data structure accessed by a data flow development tool run by a processor in a system, the data structure comprising:

nodes assigned to data processed by a data flow program and to code segments of the data flow program;

data read and data write identifiers for each code segment, the data read and data write identifiers identifying at least a portion of the data read or written by the code segment; and

dependencies between nodes determined based on the data read and data write identifiers, wherein

the development tool accesses the data structure to provide a visualization of the data flow program.

25. (Original) A computer readable memory device according to claim 24, wherein the data structure further comprises:

a processed flag that indicates whether at least one of the nodes is executed or unexecuted.

26. (Original) A computer readable memory device according to claim 24, wherein the data structure further comprises:

a taken flag that indicates whether at least one of the nodes has been claimed by a thread.

REMARKS

Claims 1-26 are pending and under consideration. Claims 1-26 were rejected in the Office Action mailed October 17, 2006. In this Response, no claims are amended, added, or canceled.

I. Obviousness Rejections under 35 U.S.C. §103(a)

Claims 1-2, 7-8, 12-13, 18-20, and 23-25 are rejected under 35 U.S.C. §103(a) as being allegedly unpatentable over *Calder et al.* (U.S. Patent No. 5,963,972, hereinafter "*Calder*") in view of *Lomet* (U.S. Patent No. 5,963,972), and further in view of *Serra et al.* (U.S. Patent No. 6,226,787, hereinafter "*Serra*"). Applicants respectfully traverse the rejection, and respectfully submit that he combination of *Calder, Lomet* and *Serra* fails to teach or suggest all of the limitations of independent claims 1, 8, 12, 19, 20, 23, and 24.

Applicants respectfully submit that there is no motivation to combine *Calder*, *Lomet,* and *Serra* as suggested by the Examiner. The Examiner asserts that the modification would be obvious because one of ordinary skill in the art would be motivated to minimize cache misses by ensuring the proper order of computer operations. However, this does not explain what benefit *Serra* provides in this hypothetical combination. The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). Because *Calder* is directed to a code reordering algorithm for optimizing a cache and does not require human input or intervention, there is no reason for displaying *Calder*'s graph to a user. Thus, modifying *Calder* with *Serra* offers no benefit, and therefore the desirability of the combination is not established.

For at least these reasons, *prima facie* obviousness has not been established, and claim 1 is patentable over the combination of *Calder*, *Lomet,* and *Serra*. Claims 8, 12, 20, 23, and 24 also recite similar limitations, and are therefore patentable for at least the same reasons.

Regarding claim 2, the combination of *Calder*, *Lomet,* and *Serra* fails to teach or suggest a graph comprising nodes assigned to the blocks and dependency arcs representing the determined dependencies. Unlike Applicants' claimed invention in which graph nodes are assigned to blocks of memory, *Calder's* graph nodes merely correspond to units of instructions -- the units of instructions do not correspond to blocks of memory. In response, the Examiner asserts that "a block is a group of instructions in a program that are treated as a unit. Therefore, the unit is store in one block of memory." The Examiner's assertion is incorrect. A block, in claim 1, is assigned both a portion of data and at least one code segment. Thus, *Calder's* assignment of nodes to units of instruction cannot be construed as an assignment of nodes to blocks of memory. Further, it is irrelevant that the unit of code instruction in *Calder* is stored in one block of memory, as alleged by the Examiner, because it is still the instruction that is assigned the node, not the block of memory. Accordingly, *Calder's* graph nodes do not correspond to blocks of memory and thus cannot teach or suggest this limitation.

For at least these reasons, *prima facie* obviousness has not been established, and claim 2 is patentable over the combination of *Calder*, *Lomet,* and *Serra*. Claim 9 recites similar limitations and is therefore patentable for at least the same reasons.

Regarding claim 19, the combination of *Calder*, *Lomet,* and *Serra* does not teach or suggest the generation and display of a graph depicting dependencies among memory regions, as previously discussed. Furthermore, the combination fails to teach or suggest "while the code segments are executing, determining which nodes in the graph are unexecuted nodes and which

nodes in the graph are executed nodes; and displaying the unexecuted nodes in a manner visually distinctive from the executed nodes." As previously discussed with regard to claim 6, *Razdow* does not visually distinguish between executed and unexecuted nodes. Accordingly, *prima facie* obviousness has not been established, and claim 19 is patentable over *Calder*, *Lomet,* and *Serra.*

Claims 3-4, 6, 9-11, 14-15, 17, and 21-22 are rejected under 35 U.S.C. §103(a) as being allegedly unpatentable over *Calder* in view of *Lomet* and *Serra*, and further in view of *Ju (U.S. Patent No. 6,175,957)*. Applicants respectfully traverse the rejection.

Claims 1, 12, and 24 are allowable as discussed above. *Ju* still fails to disclose or suggest Applicants' claimed data read and data write identifiers and fails to disclose or suggest determining dependencies based on the data read and data write identifiers. Therefore, *Calder* in view of *Lomet, Serra* and *Ju* still fails to disclose or suggest claims 1, 12, and 24.

Claims 3-4, 6, 9-11, 14-15, 17, and 21-22 depend directly or indirectly from claims 1, 12, or 24 and are therefore allowable for at least the same reasons that claims 1, 12, and 24 are allowable.

Claims 5, 16, and 26 are rejected under 35 U.S.C. §103(a) as being allegedly unpatentable over *Calder* in view of *Lomet* and *Serra*, and further in view of *Cai (U.S. Patent No. 6,349,363)*. Applicants respectfully traverse the rejection.

Claims 1, 12, and 24 are allowable as discussed above. *Cai* still fails to disclose or suggest Applicants' claimed data read and data write identifiers and fails to disclose or suggest determining dependencies based on the data read and data write identifiers. Therefore, *Calder* in view of *Lomet, Serra* and *Cai* still fails to disclose or suggest claims 1, 12, and 24.

Claims 5, 16, and 26 depend directly or indirectly from claims 1, 12, or 24 and are therefore allowable for at least the same reasons that claims 1, 12, and 24 are allowable.

CONCLUSION

In view of the foregoing, it is submitted that claims 1-26 are patentable. It is therefore submitted that the application is in condition for allowance. Notice to that effect is respectfully requested.

Respectfully submitted,

Date: April 6, 2007

A. Wesley Ferrebee
Reg. No. 51,312

Customer No. 58328
SONNENSCHEIN NATH & ROSENTHAL LLP
P. O. Box 061080
Wacker Drive Station - Sears Tower
Chicago, Illinois 60606-1080
Telephone (202) 408-6832
Facsimile: (312) 876-7934